An Online Simulation Technique for Programmed Machine Training

SIMON C.S. HUI, JAMES D.J. LIU
Industrial Centre, The Hong Kong Polytechnic University, Hong Kong, China

ABSTRACT
An online simulation technique for programmed machine training is described. Similarity of data structures with a programmed machine and online simulation software are considered to be an advantage for implementing online learning of the machine programming. JavaScript and Flash script are used to implement the simulation. In this article, it is shown how these scripts can communicate with each other to perform the functions as input, calculation and animation. Highlighted work based on a computer numerical control (CNC) milling machine is given to illustrate the working principles.

KEYWORDS
Simulation, Online, Virtual, Training, Programmed machining

INTRODUCTION
Online simulation has become a very popular technique in presenting a virtual world [1] [2] such as that in online games. Unlike other Web applications such as Web portal or e-business, online simulation offers more than dynamic data and information exchange. Simulation implies the ability to mimic real world objects or systems. It facilitates learning in an online laboratory [3].

In this paper, online simulation is applied in programmed machining. As a general description, programmed machining is an automation technology in manufacturing systems. Many machines used in the industry today are classified as Numerical Control machines under computer program control. To learn to control the machine is to familiar with the programming interface and the syntax and to observe the controlled machining process as a result of a user program [4].
This paper discusses the techniques in building an online simulation. HyperText Mark up Language (HTML) and JavaScript are necessary to develop the interface of online simulation. Macromedia’s Flash is crucial to the effect of the simulated result. To the developer, programming skill and animation experiences are required to achieve the final result [5]. In order to give a detail description of how this could be done, we introduce a machine simulation system as a working model in the following sections. In this paper, highlighted work based on a computer numerical control (CNC) milling machine is given to illustrate the working principles.

**STRUCTURE AND COMPONENTS**

To achieve the goal of a virtual machine [6], the online simulation system must provide three components: an input interface for users to input machine codes, a program window to display the program codes, and a preview window to simulate the result of the program. The first and second components are built mainly with HTML, embedded with JavaScript as the logic of the interface. The last component, on the other hand, is constructed by using the Macromedia’s Flash. Although Flash has logic script of its own, it also supports a useful technique that enables communication between Flash animation and JavaScript in a Web page. This kind of communication is very important because data exchange and cooperation are necessary between components of the system.

In the following sections, we will discuss functions as well as working process of each component. We will see how their interfaces can be constructed with HTML and how they communicate with each other. In order to explain the detail process of simulation, we will go into the core of the system and focus our discussion on the logic scripts.

**PROGRAM WINDOW**

The program window is the place where the users present their machine codes. Like all programming languages, these machine codes are generated from a command system. The command system contains the definitions of all machine functions. Each function includes function name and a set of fixed parameters so that the input of each machine code is confined in a strict order. The command system of the machine is generally a software system resided in a real machine, but in our case we will make a copy of the command system and deliver it over the Web.
Considering that there is similarity of data structures between CNC machine codes and HTML tag language, HTML is used to implement the command system.

Briefly speaking, a CNC program is written in the form of **Program Blocks**. Each Block is designed to perform a specific function, which is described by the parameters it carries. Each parameter includes name and value. If we see one program block as a single object, then its parameters would be sub-object stored in it. Such a data structure can be built in Web page using **HTML Form**. HTML Form acts as an object element that stores values in a Web page. It has sub-elements called **Input Items** that can contain all kinds of parameters depending on the types of Program Blocks.

Different kind of Program Block has different set of parameters. A list of Program Block examples is shown in Table 1 for reference. The type of the parameter and the value vary with Program Blocks. We can make a collection of HTML Forms representing all the Program Blocks with associate groups of Input Items describing parameters. The HTML Forms then become ready-to-use templates of the Program Blocks. When a user key in a function on the user interface, the associated Program Block will be displayed in the Program Window.

**Table 1  A list of Program Block examples**

<table>
<thead>
<tr>
<th>Program Block</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Program Management</strong></td>
<td>Program number</td>
</tr>
<tr>
<td><strong>Blank FORM</strong></td>
<td>spindle axis (X/Y/Z)</td>
</tr>
<tr>
<td><strong>Tool Definition</strong></td>
<td>Tool number</td>
</tr>
<tr>
<td><strong>Tool Call</strong></td>
<td>Tool number</td>
</tr>
<tr>
<td><strong>Linear Milling</strong></td>
<td>Co-ordinate set (A/I)</td>
</tr>
</tbody>
</table>
Similarly, the parameters can be divided into many types of data entries. Each type of data entry has its own rule of input and display format. All parameters are stored in the HTML Form. As a CNC part program consists of a series of program blocks entered by the user, small programs written in JavaScript in the HTML source code shall check entry syntax and store every program blocks. The Program Window continuously simulates the corresponding data entry behaviour as required by the real command system that is always scanning user’s data entry. When the program Window is asked to execute the part program, it will present the corresponding response of the machine.

In HTML code, name will be given to each form object and input item by **id** setting to each of them. The id setting will make sure we can retrieve data by their names. For example, we can retrieve a Program Block called ‘Tool Definition’ by calling id “toolDef”. We can also set id for different input elements according to what data types they stand for.

To demonstrate the way we build this data structure in HTML, one of the HTML form element taken from the simulation web page is shown below.

```html
<form id="toolDef" method="post" action="/">
  <input type="text" value='3' size="1">
  <input type="text" value='TOOL DEF' size="10">
  <input id="toolN" type="text" value='34' size="1" class="show">
  <input id="toolL" type="text" value='L ' size="3" class="show">
  <input id="toolR" type="text" value='R ' size="3" class="hide">
</form>
```

In this example we have one form object with five input items. They are all included in HTML tag and carry a number of attributes. The most important attribute of all is **id** attribute. Every HTML element can have a name after an id property so that we can retrieve its data later. The id of the FORM object indicates that all parameters stored in this form belongs to the Program Block ‘Tool Definition’. Two other properties of FORM object are method and action. They define the way of how data can be retrieved.

All input items have almost identical properties. The value attribute specifies a control’s initial value or default value, when the form is in operation, the value may be modified according to user interaction and scripts. The size property gives the limit of length of the data string. The class attribute decides whether an item should show up. Some items have id and some don’t. Any item that has an id holds a value of the parameter. Those items that have
no id hold the name of a parameter. When a Web browser processes this FORM object, all its data will be retrieved, rendered in order and the final output should look like this:

**3 TOOL DEF 34 L**

Another example below defines the Program Block ‘Tool Call’. It also has 5 parameters and all of them are coded in a Web page.

```html
<form id="toolCall" method="post" action="">
  <input type="text" value='4' size="1">
  <input type="text" value='TOOL CALL' size="11">
  <input id="toolN" type="text" value='5' size="1" class="show">
  <input id="axis" type="text" value='Z3' size="2" class="hide">
  <input id="spinSpeed" type="text" value='S2' size="2" class="hide">
  <input id="m" type="text" value='M5' size="2" class="hide">
</form>
```

And the result would be:

**4 TOOL CALL 5**

Another example defines the Program Block ‘Linear Milling’. It includes totally 8 parameters in order to describe the end point position of a line.

```html
<form id="line" method="post" action="">
  <input type="text" value='5' size="1">
  <input type="text" value='L' size="3">
  <input id="xyz" type="text" value=' x+1 ' size="4" class="show">
  <input id="xyz" type="text" value=' y+2' size="4" class="show">
  <input id="xyz" type="text" value='z-3' size="4" class="show">
  <input id="r" type="text" value='R0' size="3" class="show">
  <input id="f" type="text" value='FMAX' size="5" class="show">
  <input id="m" type="text" value='M' size="3" class="show">
</form>
```

And the rendered result should looked like this:

**5 L X+1 Y+2 Z-3 R0 FMAX M**

In Figure 1, some of the CNC codes are shown in the Program Window on the left hand side of the CNC online emulator.
PREVIEW WINDOW

The Preview Window generates interactive animation. Unlike pre-configured animation, interactive animation can create graphical results dynamically according to the event source that it receives. In our case, the event source is the data of the machine’s cutting position determined by the CNC codes at the Program Window. As shown in Figure 1, the proper graphical animation result corresponds to the part program.

The Preview Window is implemented as another frame of the CNC online emulator on the right hand side of the Program Window. In this area, users can test their CNC programs by playing the simulation to preview result of a cutting path in a graph over the Internet.

In order to achieve interactive animation, we must have a communication bridge between the Program Window and the Preview Window. Through this bridge, raw data from the CNC codes in the Program Window is transformed into graphical data such as positions and sent to the animation generator for further computation. Despite the
powerful interactive ability of Flash, it provides no advanced mathematics function. Transformation of raw data is processed in JavaScript.

In Flash script, we create a little filled circle as the Cutter. By setting the radius and x, y position of the Cutter to the calculation result, we can have, in some sense, a self-creating 2D animation. However, our simulation requires more than simply the interactive motion of the cutting head but to simulate the continuous cutting process in the machine, requiring the generation of the whole cutting path that extends every millisecond. But this kind of continuous animation is not available in Flash. We use another method called “duplication”. In this method, we simply duplicate the Cutter and move it a very small step ahead each time during the cutting process. Since each filled circle offsets from its previous position by a tiny value, the overall result would be a smooth cutting path with certain width and round head at each side. Using this method, all shapes of cutting path can be created.

The calculation of each circle’s position along the cutting path is the most mathematic intensive part of the preview model. However, Flash has relatively poor arithmetic ability, calculation is done in JavaScript. We need a special technique to allow communication between JavaScript and FLASH object. The interacting effect of the emulator will depend on the real time calculation of the simulation of machine cutting path. A user’s initial request will be sent from the Flash Object to JavaScript where the calculation of the cutting path will take place. The result will then be retrieved by the Flash object and fed for the animation generation.

The whole cutting simulation starts with the Flash animation object sending the “start” signal. As an example, the script below stays in the Flash animation when processed, calls a JavaScript function named “startCut” when processed. “FS Command ()” is the crucial command of Flash to bring a connection between Flash script and external JavaScript.

```
... FS Command ("startCut", "")
...
```

The “startCut” signal triggers some calculations and produces a series of dynamic results. These results will then be sent back to the Flash animation as positional data. The feedback is done by locating the Flash animation as an
external object of HTML. With the name space “document.Flash-Object.SetVariable()”, we are able to set value inside Flash animation crossing the HTML environment. In the following JavaScript, the variable between the parentheses named “agent” will store the value temporary for further processing.

```javascript
if (command == "startCut") {
    if (firstCut==0) {
        setValue(); firstCut=1;
    }
    eval(formula); // calculate the immediate position of cutting.
    document.FlashObject.SetVariable("/agent:x", X1);
    document.FlashObject.SetVariable("/agent:y", Y1); // set the position of the Cutter
    document.FlashObject.SetVariable("/agent:tr", 80);
    if (endOfSection==1) {
        if (parent.program.document.forms[num].id == "end") {
            document.drawline4.SetVariable("/td:stopCut", 1); // end of cutting
        } else {
            setValue(); endOfSection = 0;
        }
    }
}
```

After the required cutting path data are retrieved, Flash animation sets the XY position of a Cutter according to these data. Since the image of the cutter will remain after it is duplicated and transform, so a cutting path is continuously paved step by step. The Flash script below shows how this is done. “Set Property” is the standard command of Flash animation to set the property of an object. The object with its properties can also be duplicated by the command “Duplicate Movie Clip ()”. This Flash Script is being placed inside a cycle and loop over and over again until it receives the signal of “stopCut”.

```
Set Variable: "i" = i+1 // move a step further
Set Property ("/Cutter", X Position) = x // set x coordinate of the Cutter
Set Property ("/Cutter", Y Position) = y // set y coordinate of the Cutter
Set Property ("/Cutter", X Scale) = Radius // set the Radius of the Cutter
Set Property ("/Cutter", Y Scale) = Radius
Duplicate Movie Clip ("/Cutter", "Cutter "&i, i) // draw the cutting path by duplicating the cutter
```

**KEYBOARD WINDOW**

The Keyboard Window, as shown in Figure 2, provides an interface for users to input machine functions directly and evoke the correspondence display result. This component acts as the same role as the keyboard device in a real machine and it mimics the same event handling system. The event handling system is implemented to regard every input action as an event and searches the table stored in the system for the proper reaction. This system is done by JavaScript since JavaScript can implement a similar event handling system.
The image of the keyboard can be created in Flash animation to produce a user-friendly interface. The Flash animation also provides event handling methods of its own beside other image effects such as changing a colour with mouse over, that make Flash a wonderful tool to create interactive interface.

![Figure 2. CNC Keyboard Window](image)

When user mouse-click on the keys on the panel, signals are sent to the Program Window to trigger certain events. But the signal does not go to the Program Window directly. A transcription process is needed. The Flash object in the Keyboard Window needs an outside program to pass the signal to a JavaScript variable.

In Keyboard Window, events are received and handled by a JavaScript function Called `keyPress()`. In this function, we use `switch...case...` to recognize and deal with all kinds of possibilities when different key event values are received. As shown below, there is generally a set of handling script following every case in a command system.

```javascript
function keyPress() {
    if (addCode == -1) {addCode = window.event.keyCode;}
    switch (addCode) {
        //addCode is the value of key event
        case 32:  genSource();
        case 40:  down(); if (num>4) clear(num-4); else clear(0); oTail=0; break; //down
        case 38:  up(); if (num>4) clear(num-4); else clear(0); oTail=0; break;  //up
        ...
    }
}
```
CONCLUSIONS

An introduction to an online simulation technique is given. Online simulation can be applied in learning interactively to use a programmed machine such as a CNC machine. Practicing programming skill and experimenting results without the proximity of an expensive physical machine are enhanced over the Internet, unbounded to time, space and capacity. The fact that programmed machining relies on software technology, allows one to simulate the program interface in interactive graphics and to obtain program results according to user input.

The technique discussed in this paper illustrates how online simulation technology can be applied in programmed machining. Although the development of such a system required skilled programmer and knowledge of animation, it is practical to bring any real programmed machine virtually online. This simulation model is already being used in a training platform to provide virtual training in CNC part programming [7]. In future remote control techniques can be combined with this technology to provide the possibility of operating a real machine online for real physical machining.

REFERENCES


BIOGRAPHIES

Simon C.S. Hui received his MSc degree in Manufacturing Systems Engineering from The University of Warwick. He is now Engineer-in-charge of the Control & Informatics Unit of the Industrial Centre, The Hong Kong Polytechnic University. His main interest is technology-based training with an emphasis on online learning and hypermedia in engineering education.

James D.J. Liu is a Research Assistant in the Industrial Centre, The Hong Kong Polytechnic University. His main interest is hypermedia production.